On the Evaluation of X.509 Certificate Processing in Transport Layer Security Interceptors

On the evaluation of X.509 certificate processing in Transport Layer Security interceptors

Diana Gratiela Berbecaru
Politecnico di Torino, Italy
Dip. di Automatica e Informatica
diana.berbecaru@polito.it

Silvia Sisinni
Politecnico di Torino, Italy
Dip. di Automatica e Informatica
silvia.sisinni@polito.it

Matteo Simone
Politecnico di Torino, Italy
Corso Duca degli Abruzzi 24, 10129
Torino (ITALY)

***Abstract*—The Transport Layer Security (TLS) interceptors are applications running on client devices or on separate machines that filter TLS-protected traffic between two endpoints. They split the original TLS channel into two TLS channels and they might significantly impact the security obtained. They are increasingly used and installed by numerous end users or network administrators. It is necessary to assess X.509 certificate processing in TLS interceptors since flaws or problems in performing this task correctly and completely may weaken the client's communication security. We define X.509-related tests, which are divided into five categories based on which part(s) of the X.509 certificate fields or extensions get analyzed. We propose a method for automatically generating wrong, malformed, or unusual X.509 certificates (and chains) and configuration files suitable for the most common web servers, like Apache or Nginx. We deploy the generated configuration files on the TLS-aware web servers in an experimental testbed set up for testing the behavior of four selected TLS interceptors, two antivirus, and two proxy applications running on different operating systems. We report the results obtained, underlining the need to test in-depth such applications so that they would not decrease the security levels achieved by the clients.**

***Index Terms*—Web PKI, X.509 certificate, TLS interception, certificate validation**

I. Introduction

Digital X.509 certificates issued by CAs (Certification Authorities) in the frame of public key infrastructures have been long used to implement security properties in applications and security protocols. In particular, the TLS protocol [1] has driven the continuous update of the X.509 certificate format [2] to accommodate new extensions. TLS allows to set up a secure channel between a *client*, which is an application (like a web browser) and the destination *server*, such as a web server. The security parameters, such as the cryptographic keys for symmetric encryption and for integrity protection, or the algorithms employed, are negotiated at the connection establishment in the so-called TLS handshake. In this phase, the server must authenticate through an X.509v3 certificate, which is validated by the client to assess server's identity.

In the last decade, due to security, administrative, business reasons, or even governmental choices (see Sect. III), the TLS interception model has emerged. In this model, the original channel is split in *two* TLS channels where the element in between, called *TLS interceptor*, will in fact negotiate separate security parameters (keys, algorithms) with the client and server. The TLS interceptor may be an antivirus software running on the client platform, a control application, or an application proxy executed on a platform separate from the client and server (as in enterprise environments). The TLS interceptor might thus be transparent to the user and its presence, role, and impact on the security is not always fully understood by the end-users. Such tools are possibly used by millions of users, sometimes they are installed by administrators on behalf of the employees on new computers inside companies. Often, they are downloaded/purchased by users, and after installation, they remain active by default.

We address the validation of X.509 certificates into TLS interceptors (called also TLS proxies or middleboxes) placed in between an end-to-end channel to retain visibility into the network traffic. Besides X.509v3 specification [2], the CA/Browser (CA/B) forum, a consortium composed of the CAs and browser vendors define the baseline requirements [3] that indicate, among other things, some limitations or rules to follow to consider a certificate valid. For example, they impose shorter validity periods (of one year) for issued certificates, and prescribe checking extensions and revocation data. Unfortunately, this process is still subject to flaws or misinterpretation. In particular, "*problems in the X.509 certificate validation process such as accepting revoked, untrusted, or invalid certificates can cause dangerous consequences paving the way for attacks that may weaken the client's communication security*" [4]. Inconsistencies in performing correctly and/or completely certificate validation processing in the TLS proxies might potentially expose users to malicious man-in-the-middle (MITM) attacks. In presence of a middlebox, client applications have a TLS channel but the security guarantees offered could be lowered (as underlined also in [5]) or some security properties could even be lost as they strongly depend on the implementation, deployment and/or configuration of the middlebox. Several security aspects or characteristics need to be assessed in the TLS interceptors, as described below:

a. *Trust anchor(s) management*. In the TLS model, the client relies on a certificate trust store containing all the root CA certificates. This store is extremely important and must be adequately managed and protected, otherwise MITM attacks are possible. When using a TLS interceptor, a client installs a new trust anchor for the TLS interceptor certificate or its

issuer. Additionally, the private key corresponding to the TLS interceptor certificate must be adequately protected. Previous works observed that the management of the TLS interceptor's trust anchor (and keys) is subject to errors [5]. Since the root certificates of the TLS interceptors might be valid for a long time (around ten years), the TLS clients may be vulnerable to impersonation attacks when the private key is not suitably protected, e.g., when the TLS interceptor reuse the same public/private key across installations. If the middlebox stops working, such as due to an expired license, the root CA certificate remains installed on the client. Consequently, the applications relying on the (root) certificate trust store will continue to trust *any* web content signed by that certificate, which opens the door to potential MITM attacks. Thus, a side effect of TLS interceptor installation is that the root certificate added to the trust CAs store for that interception product has security implications that impact client applications beyond the TLS proxy's lifespan.

b. *X.509 certificate validation*. A TLS interceptor receiving a certificate from the server must process the certificate fields and extensions, as well as the certificate chains. This process is complex and often mistakenly executed. Even for well-known and tested TLS libraries and clients, this task is error-prone: the revocation status checking is often skipped [6] [7], various desktop and mobile web browsers behave differently when processing X.509 fields or extensions considered optional [8] because their interpretation is not mandated by the standard [2] or by the CA/B baseline requirements [3]. Some applications (especially the ones for mobiles) may even consider wrongly a certificate as valid even though mandatory fields contain wrong or malformed values, like the certificate expiration date [9].

c. *Resistance to TLS attacks*. The middleboxes might be subject to TLS attacks exploiting potential vulnerabilities in the protocol implementation, specification, or configuration on the end nodes [10] [11]. Outdated TLS proxies lacking support for safe TLS protocol versions, algorithms, or the ones running vulnerable software installations undermine the effort spent in securing the clients (e.g., the web browsers) that might even be installed properly and continuously updated [5].

In this work, we address the second point above, by defining firstly crafted sets of certificate profiles aimed to test the behavior of TLS interceptors when validating certificates. Our research has been inspired by three related works: Kuo et al. [9] created automatic tests used for assessing the certificate validation in the *mobile* browsers, while [4] specified several manual tests (in 2019) to check the behavior of some desktop TLS interception products and web browsers. Highly relevant for us were also the tests described in [5] that addressed the testing of some TLS proxy products (in 2015) for trust anchor management, certificate validation, and resistance to TLS attacks. Nevertheless, the tests in [5] are manual, while in our work, we have automatized the creation of tests through a specific procedure and scripts developed in Python. Next, we have set up an experimental testbed to apply the defined tests on selected TLS interceptors, namely, Mitmproxy (a free and open HTTPS proxy), Squid (a largely used caching proxy for Web), and two antivirus applications, namely Kaspersky Total Security, and ESET Smart Security, running on Windows 10, Ubuntu, and macOS platforms.

Organization. Sect. II presents selected related work, Sect. III explains background concepts, Sect. IV describes the tests defined to assess certificate validation in TLS proxies and the results obtained with the selected TLS interceptors. Finally, Sect. V concludes the paper and indicates future work.

II. Related Work

D. McLuskie and X. Belleken analyzed in 2018 [12] the behavior of TLS clients facing certificates whose format(s) deviate from the X.509v3 specification [2]. They individuated a set of twelve possible erroneous certificate configurations. They used mainly "unusual" values for the Distinguished Name (DN) present in any certificate (referring to the subject and the issuer of the certificate), the validity period, certificate serial number, and the digital signature applied on the certificate. They have run the tests against two TLS client configurations, namely a Windows 10 machine with Mozilla Firefox version 60 and a macOS 10 with Safari version 11.1. They showed that the two different systems used OpenSSL differently since there were no strict rules that the programmers had to follow. Moreover, they obtained different results on the two platforms for a selected set of tests. As mentioned in [4], "detecting different behavior between at least two TLS clients is considered an indication of a certificate validation error." Moreover, Wazan et al. [4] studied how some TLS interception products have reacted to misconfigurations of servers or malformed server certificates not respecting the standard. They considered proxies and antivirus tools like Avast antivirus, Kaspersky Total Security, ESET Internet Security, Squid, or Mitmproxy, measuring their behavior between 2017 and 2019. They have also extensively discussed revocation and tested the support for OCSP stapling in web browsers. Luo et al. [9] undertook a systematic and large-scale study of certificate validation mechanisms within highly popular mobile browsers. They evaluated 30 frequent browsers on two mobile OS versions and compared them with five representative desktop browsers. They compiled a test suite containing 157 test cases (divided into five categories) by understanding the rules of multiple RFC standards and CA/B requirements [3]. Then, they designed and implemented an automated and generic testing pipeline for mobile browsers.

III. TLS interception and X.509 processing

a) TLS interception: The TLS protocol allows to establish a secure communication channel over a TCP connection (though variants exists over UDP as well) by guaranteeing important security features. It is widely exploited in different contexts or use cases nowadays, such as digital identity management frameworks [13] [14], time distribution networks [15] [16], Internet of Things, [17], or in the supply chains.

Since the TLS traffic is encrypted between client and server, this might create problems in some application contexts or

environments. For example, the intrusion detection systems do not work properly because they have to analyze the application payload (which is hidden). Similarly, the application firewalls cannot apply firewall policies because they cannot inspect the application-level traffic. Some companies would also need retain plain traffic to demonstrate later on compliance with standards, they have to monitor outgoing traffic to prevent data loss or exfiltration, or could require access to part of the data (e.g. visited sites) to block access to some sites. Last but not least, a state government could require access to specific traffic handled via Internet Service providers.

A theoretical study of the main possible TLS interception techniques is provided in [18]. In particular, the TLS session splitting is the classical man-in-the-middle pattern, when the middlebox, acting as sever towards the client and as a client towards the server, is generally trusted by the client to verify the validity and correctness of the server certificate. This method is preferred by antivirus tools, parental control, malware, student or employee monitoring systems, adblocking software, and enterprise network appliances.

TABLE I
TESTS COVERING THE CERTIFICATE FIELDS (CLASS C1).

Test no.	Test description
T1	Leaf certificate signed with a randomly generated key instead of the intermediate CA's private key.
T2	Values of the *notBefore* and *notAfter* fields are swapped, resulting in an invalid certificate validity period.
T3	Leaf certificate without the *notBefore* value.
T4	Leaf certificate without the *notAfter* value.
T5	Leaf certificate with a long validity period (e.g., 50 years). According to [3], the certificates issued on or after 1 September 2020 should not have a validity period greater than 397 days and must not have a validity period greater than 398 days.
T6	Leaf certificate with the Subject's Common Name (SCN) set to 0x00 (NULL).
T7	Leaf certificate with the SCN set to *www.mydifferentwebsite.com* instead of *www.mywebsite.com* (i.e., DNS name of the test web server).
T8	Leaf certificate with the SCN set to the 0x09 (tab escape) value.
T9	Leaf certificate with SCN set to 0x08 (backspace escape) value.
T10	The Organization Unit Name, whose use is deprecated according to [3], contains 200 'A' characters.
T11	Leaf certificate with a 30 octets serial number, while serial number should be at most 20 octets [2].
T12	Leaf certificate with an unknown critical extension.

b) X.509 format: An X.509 certificate contains several fields, like *version*, *serial number*, *Subject*, *Issuer*, *notBefore*, *notAfter*, and extensions that are very broad in their applicability. Next, we provide a brief explanation for the main standard extensions [2] related to the defined tests.

The *Basic Constraints* (BC) extension specifies whether the subject of the certificate is a CA and the certificate chain length. If this extension is present, the pathLenConstraint (indicating the maximum number of CA certificates forming the "certificate chain"), must be greater than or equal to 0.

The *Key Usage* (KU) extension is used to define the purpose of the key contained in the certificate, more precisely, to limit

TABLE II
TESTS COVERING THE CERTIFICATE IDENTITY (CLASS C2).

Test no.	Test description
T13	Leaf certificate with SCN set to a DNS name (*www.mywebsite.com*), while the SAN extension is set to a different DNS name (i.e., *www.mydifferentwebsite.com*).
T14	Leaf certificate with SCN set to NULL (0x00) and SAN extension set to DNS name of test web server, i.e., *www.mywebsite.com*.
T15	Leaf certificate with both SCN and the SAN extension set to NULL (0x00).
T16	Leaf certificate with SCN set to DNS name of test web server (*www.mywebsite.com*), while SAN extension is absent.
T17	Leaf certificate with SCN set to 0x00, while the SAN extension of the same certificate contains two entries, namely the DNS names *www.mywebsite.com* and *www.mydifferentwebsite.com*.
T18	Leaf certificate with the SCN field set to NULL (0x00) character, while the SAN extension is not set.
T19	Leaf certificate with the SCN field set to NULL (0x00) character and the SAN extension contains a private IP address (i.e., 192.168.0.106).
T20	Leaf certificate with the SCN set to the DNS name of test web server (*www.mywebsite.com*) and the SAN extension set to a private IP address (i.e., 192.168.0.106).
T21	Leaf certificate with the SCN field set to NULL (0x00), while the SAN extension contains two entries: a DNS name (*www.mywebsite.com*) and a private IP address (i.e., 192.168.0.106).
T22	Leaf certificate with the SCN set to NULL (0x00) and the SAN extension contains a public IP address.
T23	Leaf certificate with the SCN set to the DNS name of the test web site, while the SAN extension contains its public IP address.
T24	Leaf certificate with the SCN field set to NULL (0x00), while the SAN extension contains two entries, namely a DNS name and a public IP address.
T25	Leaf certificate with SCN set to NULL (0x00), while the SAN extension contains two entries, namely the NULL (0x00) value and a public IP address.

the use of a key. In particular, a public key can be exploited for digital signature, non-repudiation, key encryption, data encipherment, and key agreement. Moreover, the CA certificates have keys for certificate signing and CRL signing. The *Extended Key Usage* (EKU) extension indicates one or more purposes for which the public key can be used, in addition to or instead of the basic ones in the KU extension, such as code-signing, S/MIME, or document signing.

The *Subject Alternative Name* (SAN) extension indicates additional identities for the subject of the certificate. In this extension, other identification information not entered in the Subject field can be used, such as a DNS name, an IP Address, an email address, or a Directory Name or an Uniform Resource Identifier (URI). The extension is often used when a certificate belongs to a host with multiple names, referred to as multi-domain certificate. As indicated in [2], multiple instances of each name form may be included in this extension.

The *CRL Distribution Points (CRLDP)* extension specifies locations from where the CRL (Certificate Revocation List) for that certificate can be obtained. A CRL is a signed list, issued periodically by the CAs, which contains the revoked certificates. The extension value can take multiple forms, it

can be an e-mail address, a URL, or an entry of one directory, although most of the time is simply an URL. The *Authority Information Access (AIA)* extension indicates the URL of the OCSP responder(s) that can be used to check the revocation status of a certificate. Moreover, it contains a URI pointing to the issuer's CA certificate. Other TLS extensions support the OCSP stapling mechanism.

TABLE III
TESTS COVERING THE CERTIFICATE REVOCATION AND CERTIFICATE CHAINS (CLASSES C3 AND C4).

Test no.	Test description
T26	Leaf certificate does not contain the URI of the intermediate CA CRL in the CRLDP extension and the OCSP URI in the AIA extension.
T27	Leaf certificate contains the CRLDP and AIA extensions but the CRL (URI) is unreachable.
T28	Leaf certificate contains the CRLDP and AIA extensions but the OCSP server is unreachable.
T29	Leaf certificate contains the CRLDP and AIA extensions. In the test, the certificate is revoked. The server is configured to send stapled certificate revocation status.
T30	Leaf certificate contains the CRLDP and AIA extensions. The test supports TLS Feature extension, a.k.a. OCSP Must-Staple. The server is configured to not send the stapled status of the certificate.
T31	Self-signed leaf certificate received by TLS interceptor.
T32	A new root CA certificate not installed in the trusted CAs store.
T33	The intermediate CA certificate has the *Basic Constraints* extension set to `CA:False`.
T34	Certificate chain with two intermediate CA certificates. The first one has the *Basic Constraints* extension set to `CA:True` and the maximum path length set to 0, meaning that that specific CA cert can issue only leaf certificates.
T35	Leaf certificate issued by a revoked intermediate CA certificate.
T36	Leaf certificate not yet valid, the validity period starts in the future (e.g., after one month).
T37	Intermediate CA certificate not yet valid, the validity period starts later (e.g., after one month).
T38	Root CA certificate not yet valid, time validity starts in the future (e.g., after one month).
T39	A chain of certificates containing also the root CA certificate is received by TLS interceptor.

IV. ANALYSIS OF SELECTED TLS INTERCEPTORS

A. Classification of tests

To assess TLS interceptor's behavior, we have defined custom sets of tests divided in the following main classes:

C1) *Certificate fields*: tests (T1-T12 in Table I) for observing TLS interceptor's behavior upon receiving a certificate not conforming to the X.509 format, or if the value(s) of the X.509 certificate fields are unknown or unrecognized. For example, a certificate which does not contain the time validity dates or with a long serial number or containing an unknown extension.

C2) *Certificate identity*: tests (T13-T25 in Table II) to observe middlebox behavior while verifying the identity of the certificate, such as checking whether the identity of the certificate is represented under the correct field or extension or (for the certificate belonging to a host registered in DNS) verifying whether hostname verification runs properly on diverse conditions (e.g., wildcard domain names). These tests exploit malformed or incorrect values for the SAN certificate extension, used in combination with the Subject field. The CA/Browser forum states that the SAN extension must be present and must contain at least one entry that could be a DNS name or a public IP address. The use of a private IP address is prohibited. Moreover, the use of the Subject's Common Name (SCN) is discouraged. If it is present, then it must contain exactly one of the entries in the SAN extension.

TABLE IV
TESTS COVERING CERTIFICATE KEY USAGE (CLASS C5).

Test no.	Test description
T40	Leaf certificate with Key Agreement (KA) set in KU extension and no EKU extension set.
T41	Leaf certificate with Data Encipherment (DE) set in KU extension and no EKU extension set.
T42	Leaf certificate with Key Encipherment (KE) set in KU extension and no EKU extension set.
T43	Leaf certificate with Digital Signature (DS) set in KU extension and no EKU extension set.
T44	Leaf certificate with KA set in KU extension and EKU extension set to server authentication.
T45	Leaf certificate with DE set in KU extension and EKU extension set to server authentication.
T46	Leaf certificate with KE set in KU extension and EKU extension set to server authentication.
T47	Leaf certificate with DS set in KU extension and EKU extension set to server authentication.
T48	Leaf certificate with DS set in KU extension and EKU extension set to client authentication.
T49	The leaf certificate has no KU extension set and EKU extension is set to client authentication.

C3) *Certificate revocation*: tests (T26-T30 in Table III) aimed to verify the processing of certificate revocation extensions (CRLDP, AIA) in the tested middlebox and TLS interceptor's behavior if the revocation data is unreachable.

C4) *Certificate chains*: tests (T31-T39 in Table III) used for evaluating TLS interceptor's behavior when processing anomalous certificate chains, e.g., chains containing intermediate CA or root CA certificates whose validity period starts in the future, or intermediate CA certificates with the BC extension set to `CA:False`, which means they should not be used to sign other (user) certificates.

C5) *Certificate key usage*: tests (T40-T49 in Table IV) used to assess the processing of KU and EKU extensions in TLS interceptors. These tests aim to identify whether the certificate follows the scope and purpose indicated in the extensions and whether there are inconsistencies between the KU field and the EKU extension.

B. Testbed description and Results

We have used the latest version of the OpenSSL library to create test certificate chains (composed of a leaf, intermediate, and root CA certificate) issued by demo CAs that allowed support for CRL and OCSP revocation mechanisms. We have set up a testbed with three desktop machines, one for the client, one for the TLS interceptor, and one for the web server.

TABLE V
SELECTED TLS INTERCEPTION PRODUCTS (TESTED IN 2022).

OS	TLS interceptor software
Ubuntu 20.04.4 LTS	Mitmproxy v8.1.1, Squid v5.5
Microsoft Windows 10	Mitmproxy v8.1.1, Squid v4.14 Kaspersky Total Security, ESET Smart Security
macOS Monterey 12.6	Mitmproxy v8.1.1, Squid v4.17

We exploited Ubuntu 20.04.4 for the server and several OS platforms (Windows, macOS, and Microsoft Windows 10) to run the selected TLS interceptors shown in Table V. In some tests, instead of three physical machines, we have used virtual machines on two physical machines since this choice does not impact the test results. In the tests with Mitmproxy, each client runs the Firefox browser, which is configured to connect to the server through the (IP address of) machine running the Mitmproxy (listening on port 8080). In the tests with macOS the proxy is installed directly on the physical machine acting as a client too. In the case of Squid, we used a similar testbed as the one used for Mitmproxy, except for the proxy application and the port used, i.e., port 3128. As antivirus products, we installed Kaspersky Total Security and ESET Smart Security on a Windows 10 machine, acting as a client.

To automate and simplify the generation of all the certificates together with server configurations for common web servers, we implemented a set of Python scripts that combine the use of two Python libraries, namely PyOpenSSL v.22[1] and cryptography v.39[2]. The testing software is modular and extendable. It allowed us to build the testing material automatically in a few steps: we configured firstly the certificate profiles containing the values for certificate fields and extensions as indicated in the tests' description. Then, we developed scripts in Python for creating the corresponding certificates and chains. Finally, another script created the configuration files ready to be deployed on the web server used in the testbed, e.g., Apache, Nginx, or Lighttpd.

Results discussion. By analyzing the results of the tests T1-T12 in Table VI, we observe that Mitmproxy and Squid correctly rejected malformed certificates, but considered valid the certificates in T10 and T11. Kaspersky instead opted for a soft-fail, i.e., a warning displayed to the user in the majority test cases, while ESET delegated the validation of certificates to the client (i.e., to Mozilla Firefox browser) in T2, T6, T7, T8, and T9. Regarding the processing of the SAN extension, the behavior of the TLS interceptors diverges, as shown in the tests T13-T25 in Table V. Mitmproxy blocked all the TLS connections (on all platforms) except in the tests T14, T17, T21, and T24 when it generated its own certificates and delegated the validation to the client resulting in a warning message shown to the end user. On the contrary, Squid proxy recognized as valid the certificates in the above tests. It delegated instead the certificate validation to the client in the tests T13, T16, T20, and T23. In all the other tests, it blocked the connection.

Kaspersky considered valid all the certificates in which just one among the Subject CN and the SAN entries had a valid value, and rejected the connection (hard-fail) in the test T25. ESET Smart Security considered valid only the certificates that had at least one SAN entry configured properly. In all the other cases, it generated a certificate with its self-signed certificate and delegated the validation to the client which showed a warning message (in each test case). None of the

[1]https://www.pyopenssl.org/en/22.0.0/

[2]https://pypi.org/project/cryptography/39.0.0/

TABLE VI
TEST RESULTS.

Test no.	macOS		Windows				Ubuntu	
	Mitmproxy	Squid	Mitmproxy	Squid	Kasperky	ESET	Mitmproxy	Squid
T1	R	R	R	R	W	W	R	R
T2	R	R	R	R	W	→	R	R
T3	-	-	-	-	-	-	-	-
T4	-	-	-	-	-	-	-	-
T5	✓	✓	✓	✓	✓	✓	✓	✓
T6	R	R	R	R	W	→	R	R
T7	R	R	R	R	W	→	R	R
T8	R	R	R	R	W	→	R	R
T9	R	R	R	R	W	→	R	R
T10	✓	✓	✓	✓	✓	✓	✓	✓
T11	✓	✓	✓	✓	✓	✓	✓	✓
T12	R	R	R	R	W	W	R	R
T13	R	→	R	→	W	→	R	→
T14	→	✓	→	✓	✓	✓	→	✓
T15	R	R	R	R	R	→	R	R
T16	R	→	R	→	✓	→	R	→
T17	→	✓	→	✓	✓	✓	→	✓
T18	R	R	R	R	W	→	R	R
T19	R	R	R	R	W	→	R	R
T20	R	→	R	→	✓	→	R	→
T21	→	✓	→	✓	✓	✓	→	✓
T22	R	R	R	R	W	→	R	R
T23	R	→	R	→	✓	→	R	→
T24	→	✓	→	✓	✓	✓	→	✓
T25	R	R	R	R	R	→	R	R
T26	✓	✓	✓	✓	✓	✓	✓	✓
T27	✓	✓	✓	✓	✓	✓	✓	✓
T28	✓	✓	✓	✓	✓	✓	✓	✓
T29	✓	✓	✓	✓	✓	R	✓	✓
T30	R	R	R	R	W	W	R	R
T31	R	R	R	R	W	W	R	R
T32	R	R	R	R	W	W	R	R
T33	R	R	R	R	R	W	R	R
T34	R	R	R	R	R	W	R	R
T35	✓	✓	✓	✓	✓	R	✓	✓
T36	R	R	R	R	W	→	R	R
T37	R	R	R	R	W	W	R	R
T38	R	R	R	R	W	W	R	R
T39	R	✓	✓	✓	✓	✓	✓	✓
T40	✓	✓	✓	✓	✓	✓	✓	✓
T41	R	R	R	R	✓	✓	R	R
T42	✓	✓	✓	✓	✓	✓	✓	✓
T43	✓	✓	✓	✓	✓	✓	✓	✓
T44	✓	✓	✓	✓	✓	✓	✓	✓
T45	R	R	R	R	✓	✓	R	R
T46	✓	✓	✓	✓	✓	✓	✓	✓
T47	✓	✓	✓	✓	✓	✓	✓	✓
T48	R	R	R	R	W	W	R	R
T49	R	R	R	R	R	W	R	R

Legend: ✓: certificate considered valid; W: Warning shown to the user, connection continued; -: Web server not started (certificate format error); R: Connection rejected. →: certificate validation delegated to the client.

TLS interception products rejected the connection if they were not able to retrieve certificate revocation information. All of them but ESET showed a lack of support for OCSP stapling since they accepted all revoked certificates. ESET, on the other hand, correctly recognized when the certificate was revoked and blocked the connection. All of them did not support OCSP Must-Staple but rejected the TLS connection since it was considered an unrecognized X.509 critical extension. The last set of tests has shown that the TLS interceptors were not able to recognize that a certificate was issued by a revoked CA certificate. Except for Kaspersky, the tested TLS interceptors did not download the CRLs or the CA certificates from the stores available respectively in the CRLDP and in the AIA extensions. Even when the CRL was downloaded in T35 this should have caused the rejection of the connection, but this did not happen.

In the tests T31-T38 (except T35), Mitmproxy and Squid blocked all the connections. In the test T39 in which the server provided the certificate chain plus the root CA certificate no error occurred, except for Mitmproxy running on macOS where T39 resulted in a rejection of the connection. Kaspersky let server certificates pass the validation process in T35 and T39, while T33 and T34 resulted in a hard-fail (rejection) of the connection. The remaining tests in the set generated a soft-fail of the connection. ESET always opted for a soft-fail of the connection but for T36 when we obtained a delegated validation, which resulted in a warning message shown by the client. Also, ESET has also recognized as valid a chain containing the root CA certificate, and it showed an error message blocking the connection for T35.

In the tests T40-T49, Mitmproxy and Squid accepted as valid all the certificates with KU different from Data Encipherment (DE) and those in which the EKU was set to client authentication. Kaspersky and ESET soft-failed the connection only when the EKU of the certificate was set to client authentication. There were no differences in the results obtained with Lighttpd. Nginx showed a different behavior for ESET when testing OCSP stapling because it considered valid a revoked certificate or a certificate issued by a revoked intermediate CA certificate in the tests T29 and T35.

V. Conclusions and future work

TLS interceptors act between the client and servers and could (if wrongly implemented or configured) significantly degrade or even cancel the security offered by the servers and supported in client applications, e.g. in web browsers. Thus, we explain the necessity to test them thoroughly. We have selected some TLS interceptors and tested certificate processing via an automatic testing procedure with different certificate profiles, measuring their behavior. Since the standards are not very strict for some X.509 fields and extensions, we obtained indeed different results among the considered products. This study could be further extended with an analysis of trust anchor management and testing of the robustness of TLS interceptors to TLS attacks.

Acknowledgments. Dr. Diana Gratiela Berbecaru carried out this study within the Ministerial Decree no. 1062/2021 and received funding from the FSE REACT-EU - PON Ricerca e Innovazione 2014-2020. Matteo Simone performed his work during preparation of his graduation thesis at Politecnico di Torino. This manuscript reflects only the authors' views, findings, conclusions, and opinions, neither the European Union nor the European Commission can be considered responsible for them.

References

[1] E. Rescorla, "The Transport Layer Security (TLS) Protocol: Version 1.3," RFC 8446.

[2] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile," RFC-5280, May 2008, doi: 10.17487/RFC5280

[3] CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates", Version 1.8.6, 14 Dec. 2022.

[4] A. S. Wazan et al., "On the Validation of Web X.509 Certificates by TLS Interception Products," in IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 1, pp. 227-242, 1 Jan.-Feb. 2022, doi: 10.1109/TDSC.2020.3000595.

[5] X. de Carné de Carnavalet and M. Mannan, "Killed by Proxy: Analyzing Client-end TLS Interception Software," NDSS'16, Feb. 21-24, 2016, San Diego (CA, USA), doi: 10.14722/ndss.2016.23374

[6] D.G. Berbecaru and A. Lioy, "An Evaluation of X.509 Certificate Revocation and Related Privacy Issues in the Web PKI Ecosystem," in *IEEE Access*, vol. 11, pp. 79156-79175, 2023, doi: 10.1109/ACCESS.2023.3299357.

[7] C. Brubaker et al., "Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations," 2014 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 2014, pp. 114-129, doi: 10.1109/SP.2014.15.

[8] J. Larisch et al., "Hammurabi: A Framework for Pluggable, Logic-Based X.509 Certificate Validation Policies," in Proc. of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), pp. 1857-1870, doi: 10.1145/3548606.3560594.

[9] M. Luo et al. "On the Complexity of the Web's PKI: Evaluating Certificate Validation of Mobile Browsers," in IEEE Transactions on Dependable and Secure Computing, vol. 21, no. 1, pp. 419-433, Jan.-Feb. 2024, doi: 10.1109/TDSC.2023.3255869.

[10] M. Maehren et al. "TLS-Anvil: Adapting Combinatorial Testing for TLS Libraries," 31st USENIX Security Symposium (USENIX Security 22), USENIX Association, Boston, MA, 2022.

[11] D.G. Berbecaru and G. Petraglia, "TLS-Monitor: A Monitor for TLS Attacks," 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2023, pp. 1-6, doi: 10.1109/CCNC51644.2023.10059989.

[12] D. McLuskie and X. Belleken, "X.509 Certificate Error Testing," ARES 2018: Proc. of the 13th International Conference on Availability, Reliability and Security, Aug. 2018, pp. 1–8, doi: 10.1145/3230833.3232820.

[13] D.G. Berbecaru, A. Lioy and C. Cameroni, "Providing Login and Wi-Fi Access Services With the eIDAS Network: A Practical Approach," in *IEEE Access*, vol. 8, pp. 126186-126200, 2020, doi: 10.1109/ACCESS.2020.3007998.

[14] D. G. Berbecaru, A. Lioy and C. Cameroni, "On Enabling Additional Natural Person and Domain-Specific Attributes in the eIDAS Network," in *IEEE Access*, vol. 9, pp. 134096-134121, 2021, doi: 10.1109/ACCESS.2021.3115853.

[15] M. Pini et al., "Satellite-derived Time for Enhanced Telecom Networks Synchronization: the ROOT Project," in 2021 IEEE 8th Intl. Workshop on Metrology for AeroSpace (MetroAeroSpace), Naples, Italy, 2021, pp. 288-293, doi: 10.1109/MetroAeroSpace51421.2021.9511780.

[16] D.G. Berbecaru et al., "Mitigating Software Integrity Attacks With Trusted Computing in a Time Distribution Network," in *IEEE Access*, vol. 11, pp. 50510-50527, 2023, doi: 10.1109/ACCESS.2023.3276476.

[17] D.G. Berbecaru and L. Pintaldi, "Exploiting Emercoin Blockchain and Trusted Computing for IoT Scenarios: A Practical Approach," 2023 IEEE Symposium on Computers and Communications (ISCC), Gammarth, Tunisia, 2023, pp. 771-776, doi: 10.1109/ISCC58397.2023.10217961.

[18] XC. de Carnavalet and P. C. van Oorschot, "A survey and analysis of TLS interception mechanisms and motivations," 2020, doi: 10.48550/ARXIV.2010.16388

[19] S. Santesson et al., "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP," RFC-6960, June 2012.